ISC EXAMINATION PAPER – 2025
COMPUTER SCIENCE
PAPER – 1
Class – 12th
(Solved)

Maximum Marks: 70 *Time Allotted: Three Hours* *Reading Time: Additional Fifteen Minutes*

Instructions to Candidates:

1. You are allowed an **additional fifteen minutes** for **only** reading the question paper.
2. You must **NOT** start writing during the reading time.
3. This question paper has **15 printed pages and one blank page.**
4. It is divided into **two parts: Part I** and **Part II.**
5. It has **11 questions** in all.
6. **Part I** is compulsory and has **two** questions.
7. While attempting **Multiple Choice Questions** in **Part I,** you are required to **write only ONE option as the answer.**
8. **Part II** is divided into **three sections: A, B and C.**
9. **Each section in Part II** has **three questions. Any two** questions have to be attempted from each section.
10. The intended marks for questions are given in brackets [].

PART I – 20 MARKS

*Answer **all** questions.*

While answering questions in this Part, indicate briefly your working and reasoning, wherever required.

Question 1

(i) The complement of the Boolean expression $(A \cdot B') + (B' \cdot C)$ is: **[1]**

(a) $(A + B') \cdot (B' + C)$ **(b)** $(A' \cdot B) + (B \cdot C')$

(c) $(A' + B) \cdot (B + C')$ **(d)** $(A \cdot B') + (B \cdot C')$

(ii) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. **[1]**

Assertion: The expression $\sim (X \vee Y)$ is logically equivalent to $(\sim X \wedge \sim Y)$

Reason: The commutative property of logical operators states that the order of the operands does not change the result of a binary operation.

(a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.

(b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

(iii) According to the Principle of Duality, the Boolean equation **[1]**
$(1 + Y) \cdot (X + Y) = Y + X'$ will be equivalent to:

(a) $(1 + Y') \cdot (X' + Y') = Y' + X$

(b) $(0 \cdot Y) + (X \cdot Y) = Y \cdot X'$

(c) $(0 + Y) \cdot (X + Y) = Y + X'$

(d) $(1 \cdot Y) + (X \cdot Y) = Y \cdot X'$

(iv) The Associative Law states that: **[1]**

(a) $A \cdot B = B \cdot A$

(b) $A + B = B + A$

(c) $A \cdot (B + C) = A \cdot B + A \cdot C$

(d) $A + (B + C) = (A + B) + C$

(v) Consider the following code statement: **[1]**

```
public class Person
{ int age;
   public Person (int age)
   {
      this.age = age;
   }
}
```

Which of the following statements are valid for the given code?

I. The keyword *this* in the constructor refers to the current instance of the class.

II. The keyword *this* differentiates between the instance variable age and the parameter age.

III. The keyword *this* can be used only in constructors.

(a) Only I and II **(b)** Only II and III

(c) Only I and III **(d)** Only III

(vi) Given below are two statements marked, Assertion and Reason. Read the two statements carefully and choose the correct option. **[1]**

Assertion: The break statement prevents fall through effect in switch case construct.

Reason: The break statement enables unnatural exit from the loop.

(a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.

(b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

(vii) The canonical expression of F(P, Q, R) = $\pi(2, 5, 7)$ is: **[1]**

(a) $(P + Q' + R) \cdot (P' + Q + R') \cdot (P' + Q' + R')$

(b) $(P \cdot Q' \cdot R) + (P' \cdot Q \cdot R') + (P' \cdot Q' \cdot R')$

(c) $(P' + Q + R') \cdot (P + Q' + R) \cdot (P + Q + R)$

(d) $(P' \cdot Q \cdot R') + (P \cdot Q' \cdot R) + (P \cdot Q \cdot R)$

(viii) Study the given propositions and the statements marked, Assertion and Reason that follow it. Choose the correct option on the basis of your analysis. **[1]**

P – It is a holiday

Q – It is a Sunday

Assertion: If it is not a Sunday, then it is not a holiday. (Q′ => P′)

Reason: Inverse is formed when antecedent and consequent are interchanged.

(a) Both Assertion and Reason are true and Reason is the correct explanation for Assertion.

(b) Both Assertion and Reason are true but Reason is not the correct explanation for Assertion.

(c) Assertion is true and Reason is false.

(d) Both Assertion and Reason are false.

(ix) For the given code segment, write Big O notation for worst case complexity. **[1]**

```
for (int i = I; i <= P; i++)
  { Statements }
    for (int j = 1; j <= P; ++j )
      for (int k = I; k <= Q; k++)
        { Statements }
```

(x) Write the minterms in canonical form for the Boolean Function X (A, B), from the truth table given below: **[1]**

A	B	X
0	0	1
0	1	0
1	0	0
1	1	1

Question 2

(i) Convert the following infix notation to postfix form. **[2]**

(A − B) / C) + (D * E / F) * G

(ii) A matrix M [−1...10, 4....13] is stored in the memory with each element requiring 2 bytes of storage. If the base address is 1200, find the address of M[2][7] when the matrix is stored **Row Major Wise.** **[2]**

(iii) The following function *int solve()* is a part of some class. Assume 'm' and 'n' are positive integers. Answer the questions given below with dry run/ working.

```
int solve(int m, int n)
  {
    int k = 1;
    if(m < 0)
      return − k;
    else if(m==0)
```

```
        return m;
    else
        return k+(solve(m−n, n+2));
  {
```

(a) What will the function **solve**() return if: **[2]**

(1) m = 16, n = 1

(2) m = 9, n = 1

(b) What is the function **solve**() performing apart from recursion? **[1]**

(iv) The following function **duck**() is a part of some class which is used to check if a given number is a duck number or not. There are some places in the code marked by **?1?, ?2?, ?3?** which may be replaced by a statement / expression so that the function works properly.

A number is said to be Duck if the digit zero (0) is present in it.

```
boolean duck(int a)
{ int f = - 1;
  if(a==0)
    return true;
for(int i=a; i!=0;?1?)
{ int c = 1%10;
  if(c==?2?)
    { f = 1; break; }
}
return (f==?3?)? false true:
}
```

(a) What is the expression or statement at **?1?** **[1]**

(b) What is the expression or statement at **?2?** **[1]**

(c) What is the expression or statement at **?3?** **[1]**

PART II – 50 MARKS

*Answer **six** questions in this part, choosing **two** questions from Section A, **two** from Section B and **two** from Section C.*

SECTION – A

*Answer **any two** questions.*

Question 3

(i) A superhero is allowed access to a secure Avengers facility if he / she meets any of the following criteria: **[5]**

- The superhero has Avengers' membership and possesses a high-security clearance badge

OR

- The superhero does not have Avengers membership but holds a special permit issued by S.H.I.E.L.D. along with a high-security clearance badge

OR

- The superhero is not a recognised ally but holds a special permit issued by S.H.I.E.L.D. along with a high-security clearance badge

The inputs are:

INPUTS	
A	Superhero has Avengers membership.
S	Superhero holds a special permit issued by S.H.I.E.L.D.
C	Superhero possesses a high-security clearance badge
L	Superhero is a recognised ally

(In all the above cases, 1 indicates YES and 0 indicates NO)

Output: X – Denotes allowed access [I indicates YES and 0 indicates NO in all cases]

Draw the truth table for the inputs and outputs given above. Write the **POS** expression for **X (A, S, C, L).**

(ii) Reduce the above expression **X (A, S, C, L)** by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs). **[5]**

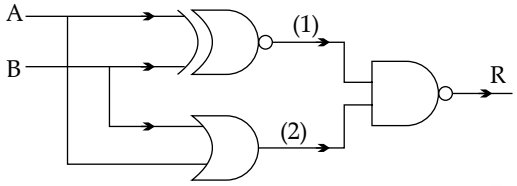
Draw the logic gate diagram using **NOR** gates only for the reduced expression. Assume that the variables and their complements are available as inputs.

Question 4

(i) (a) Reduce the Boolean function **F(P, Q, R, S)** = ∑(0, 1, 2, 5, 7, 8, 9, 10, 13, 15) by using 4-variable Karnaugh map, showing the various groups (i.e., octal, quads and pairs). **[4]**

(b) Draw the logic gate diagram using **NAND** gates only for the reduced expression. Assume that the variables and their complements are available as inputs. **[1]**

(ii) From the logic gate diagram given below:



(a) Derive Boolean expression for (1), (2) and R. Reduce the derived expression. **[4]**

(b) Name the logic gate that represents the reduced expression. **[1]**

Question 5

(i) What is an *encoder*? Draw the logic gate diagram for an octal to binary encoder. State one application of a *decoder*. **[5]**

(ii) By using truth table, verify if the following proposition is valid or not. **[3]**

$(\sim X \Rightarrow Y) \wedge X = (X \wedge \sim Y) \vee (X \wedge Y)$

(iii) Study the logic gate diagram given below and answer the questions that follow:

What will be the output of the above gate when:

(a) A = 1, B = 0 **[1]**

(b) A = 1, B = 1 **[1]**

SECTION – B

*Answer **any two** questions.*

Each program should be written in such a way that it clearly depicts the logic of the problem.

This can be achieved by using mnemonic names and comments in the program.

(Flowcharts and Algorithms are **not** required.)

The programs must be written in Java.

Question 6

A class **Perni** has been defined to accept a positive integer in binary number system from the user and display if it is a Pernicious number or not.

[A pernicious number is a binary number that has minimum of two digits and has prime number of 1's in it.]

Examples:

- 101 is a pernicious number as the number of 1's in 101 = 2 and 2 is prime number.
- 10110 is a pernicious number as the number of 1's in 10110 3 = and 3 is prime number.
- 1111 is a **NOT** a pernicious number as the number of 1's in 1111 = 4 and 4 is **NOT** a prime number.

The details of the members of the class are given below:

Class name	:	**Perni**
Data member/instance variable:		
num	:	to store a binary number
Methods/Member functions:		
Perni()	:	constructor to initialise the data member with 0
void accept()	:	to accept a binary number (containing 0's and 1's only)
int countOne(int k)	:	to count and return the number of 1's in 'k' using **recursive technique**
void check()	:	to check whether the given number is a pernicious number by invoking the function **countOne**() and to display an appropriate message

Specify the class **Perni** giving the details of the **constructor**(), **void accept**(), **int countOne**(int) and **void check**(). Define a **main**() function to create an object and call the functions accordingly to enable the task.

Question 7 **[10]**

Design a class **Colsum** to check if the sum of elements in each corresponding column of two matrices is equal or not. Assume that the two matrices have the same dimensions.

Example:

Input:

MATRIX A

2	3	1
7	5	6
1	4	2

MATRIX A

7	4	2
1	3	1
2	5	6

Output: Sum of corresponding columns is equal.

The details of the members of the class are given below:

Class name	:	**Colsum**
Data members/instance variables:		
mat[] []	:	to store the integer array elements
m	:	to store the number of rows
n	:	to store the number of columns
Member functions/methods:		
Colsum(int mm, int nn)	:	parameterised constructor to initialise the data members m=mm and n = nn
void readArray()	:	to accept the elements into the array
boolean check(Colsum A, Colsum B)	:	to check if the sum of elements in each column of the objects A and B is equal and return true otherwise, return false
void print()	:	to display the array elements

Specify the class **Colsum** giving details of the **constructor(int, int), void readArray**(), **boolean check**(Colsum, Colsum), and **void print**(). Define the **main**() function to create objects and call the functions accordingly to enable the task.

Question 8 **[10]**

A class **Flipgram** has been defined to flip the letters of the left and right halves of a **non-heterogram** word. If the word has odd number of characters, then the middle letter remains at its own position.

A **heterogram** is a word where no letter appears more than once.

Example 1: INPUT: BETTER
OUTPUT: TERBET

Example 2: INPUT: NEVER
OUTPUT: ERVNE

Example 3: INPUT: THAN
OUTPUT: HETEROGRAM

The details of the members of the class are given below:

Class name	:	**Flipgram**
Data members/instance variables:		
word	:	to store a word
Methods/Member functions:		
Flipgram(String s)	:	parameterised constructor to assign word = s
boolean ishetero()	:	to return true if word is a heterogram else return false
String flip()	:	to interchange the left and right sides of a non-heterogram word and return the resultant word
void display()	:	to print the flipped word for a non-heterogram word by invoking the method **flip**(). An appropriate message should be printed for a heterogram word

Specify the class **Flipgram** giving the details of the **constructor(String), boolean ishetero**(), **String flip**() and **void display**(). Define a **main**() function to create an object and call the functions accordingly to enable the task.

SECTION – C

*Answer **any two** questions.*

Each program should be written in such a way that it clearly depicts the logic of the problem stepwise.
This can be achieved by using comments in the program and mnemonic names or pseudo codes for algorithms.
The programs must be written in Java and the algorithms must be written in general standard form, wherever required/specified.
(Flowcharts are **not** required.)

Question 9

A circular queue is a linear data structure that allows data insertion at the rear and removal from the front, with the rear end connected to the front end forming a circular arrangement.
The details of the members of the class are given below:

Class name	:	**CirQueue**
Data members/instance variables:		
Q[]	:	array to hold integer values
cap	:	maximum capacity of the circular queue
front	:	to point the index of the front
rear	:	to point the index of the rear
Methods/Member functions:		
CirQueue(int n)	:	constructor to initialise cap = n, front = 0 and rear = 0
void push(int v)	:	to add integers from the rear index if possible else display the message "QUEUE IS FULL"
int remove()	:	to remove and return the integer from front if any, else return −999
void print()	:	to display the elements of the circular queue in the order of front to rear

(i) Specify the class **CirQueue** giving the details of the functions **void push(int)** and **int remove**(). Assume that the other functions have been defined. **[4]**

The main() function and algorithm need NOT be written.

(ii) State *one* application of a circular queue. **[1]**

Question 10 **[5]**

A superclass **Flight** has been defined to store the details of a flight. Define a subclass **Passenger** to calculate the fare for a passenger.
The details of the members of both the classes are given below:

Class name	:	**Flight**
Data members/instance variables:		
flightno	:	to store the flight number in string
dep_time	:	to store the departure time in string
arr_time	:	to store the arrival time in string
basefare	:	to store the base fare in decimal
Methods/Member functions:		
Flight(...)	:	parameterised constructor to assign values to the data members
void show()	:	to display the flight details
Class name	:	**Passenger**
Data members/instance variables:		
id	:	to store the ID of the passenger
name	:	to store the name of the passenger
tax	:	to store the tax to be paid in decimal
tot	:	to store the total amount to be paid in decimal
Methods/Member functions:		
Passenger(...)	:	parameterised constructor to assign values to the data members of both the classes

void cal()	:	to calculate the tax as 5% of base fare and total amount (base fare + tax)
void show()	:	to display the flight details along with the passenger details and total amount to be paid

*Assume that the super class **Flight** has been defined*. Using the **concepts** of **Inheritance**, specify the class **Passenger** giving the details of **constructor**(...). **void cal**() and **void show**().

The super class, main function and algorithm need NOT be written.

Question 11

(i) A linked list is formed from the objects of the class **Cell**. The class structure of the Cell is given below: **[2]**

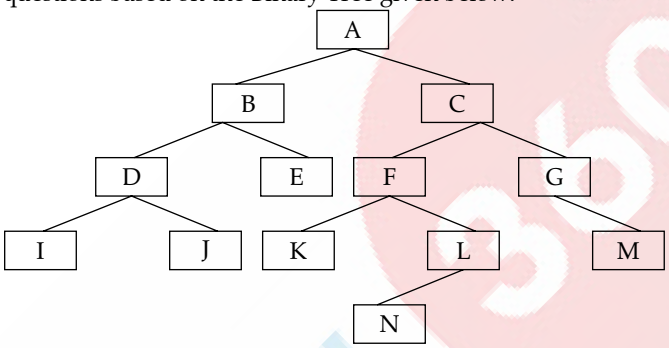
```
class Cell
{
   char m;
   Cell right;
}
```

Write an *Algorithm* **OR** a *Method* to print the sum of the ASCII values of the lower case alphabets present in the linked list.

The method declaration is as follows:

void lowercase(Cell str)

(ii) Answer the following questions based on the Binary Tree given below:



(a) Write the in-order traversal of the right subtree. **[1]**

(b) State the depth of the entire binary tree and depth of node E. **[1]**

(c) Name the external nodes of the left subtree and internal nodes of the right subtree. **[1]**

ANSWERS

PART – I

Answer 1.

(i) Option (c) is correct.

Explanation: Complement of an expression means, each and every term and the operator has to be represented in their complement form. Use DeMorgan's theorem to find the complement of the expression.

$(A \cdot B') + (B' \cdot C)$

$= ((A \cdot B') + (B' \cdot C))'$

$= (A' + B) \cdot (B + C')$

(ii) Option (b) is correct.

Explanation: $\sim(X \vee Y) = (\sim X \wedge \sim Y)$. This is a valid preposition under the DE Morgan's Law. Commutative Property states that

$X \vee Y = Y \vee X$

$X \wedge Y = Y \wedge X$

Hence, assertion and reason were true but reason does not correctly explain the assertion.

(iii) Option (b) is correct.

Explanation: Principle of Duality states that if the operator or values change, the equation remains valid or true only. To get the dual expression, the following changes take place:

$+ \rightarrow \cdot$

$\cdot \rightarrow +$

$0 \rightarrow 1$

$1 \rightarrow 0$

$(1 + Y) \cdot (X + Y) = Y + X'$

$= (0 \cdot Y) + (X \cdot Y) = Y \cdot X'$

(iv) Option (d) is correct.

Explanation: The Associative Law states that the grouping of variables does not affect the result of the operation. It applies to both AND (·) and OR (+) operations.

(v) Option (a) is correct.

Explanation: The 'this keyword' refers to the current instance of the class in which it is used. The 'this keyword' is used to distinguish between instance variables and parameters with the same name. The 'this keyword' can be used in constructors, methods and even to call other constructors within the same class (this()).

(vi) Option (b) is correct.

Explanation: The break statement stops the execution of a switch case and prevents control from falling through to the next case. Without a break, the execution continues into the next case, causing the fall-through effect. The break statement is also used in loops (for, while, do-while) to exit before the loop condition fails. It allows an early exit, which is considered an "unnatural" exit since it does not follow the normal loop termination condition. Hence, both the assertion and reason were true but reason does not correctly explain the assertion.

(vii) Option (a) is correct.

Explanation: Convert the given Maxterms to expressions:

F (P, Q, R) = Π (2, 5, 7)

M2 = (P + Q' + R)

M5 = (P' + Q + R')

M7 = (P' + Q' + R')

Final canonical expression is: (P + Q' + R). (P' + Q + R'). (P' + Q' + R')

(viii) Option (d) is correct.

Explanation: $Q \rightarrow P$ (If it is a Sunday, then it is a holiday)

The contrapositive of $Q \rightarrow P$

$\neg P \rightarrow \neg Q$ (If it is not a holiday, then it is not a Sunday)

Given assertion is:

$\neg Q \rightarrow \neg P$ (If it is not a Sunday, then it is not a holiday)

Hence, the assertion is false.

The inverse of $Q \rightarrow P$ is: $\sim Q \rightarrow \sim P$

The inverse is actually formed by negating both antecedent and consequent, not by interchanging them.

Interchanging antecedent and consequent forms the converse ($P \rightarrow Q$), not the inverse.

Hence, both the assertion and reason are false.

(ix) First Loop runs for O(P)

Second loop is the nested loops have complexity $O(P \times Q)$

Since $O(P \times Q)$ dominates O(P) when Q>=1, the overall worst time complexity is $O (P \times Q)$

(x) m0 = A'B'

m3 = AB

Final expression is: A'B' + AB

Answer 2.

(i) (A − B/C) + (D * E/F) * G

= A B C / − + (D * E/F)*G

= A B C / − D E * F / * G

= A B C / − D E * F / G *

= A B C / − D E * F / G * +

(ii) Address(M[2,7]) = 1200+[(2−(−1))×10+(7−4)]×2
= 1200+[(2+1)×10+3]×2
= 1200+[3×10+3]×2
= 1200+[30+3]×2
=1200 + 33 × 2
= 1200 + 66
= 1266

(iii) (a) (1) m = 16, n = 1

Call	m	n	solve (m, n)
solve(16, 1)	16	1	1 + solve(15, 3)
solve(15, 3)	15	3	1 + solve(12, 5)
solve(12, 5)	12	5	1 + solve(7, 7)
solve(7, 7)	7	7	1 + solve(0, 9)
solve(0, 9)	0	9	returns 0

Result: 1 + 1 + 1+ 1 = 4

(2) m = 9, n = 1

Call	m	n	solve(m, n)
solve(9, 1)	9	1	1 + solve(8, 3)
solve(8, 3)	8	3	1 + solve(5, 5)
solve(5, 5)	5	5	1 + solve(0, 7)
solve(0, 7)	0	7	returns 0

Result: 1 + 1 + 1 = 3

(b) Counting the number of steps required to reduce m to 0 by subtracting increasing odd numbers (n, n + 2, n + 4, ...).

It acts as a step counter following an arithmetic pattern.

(iv) (a) i/ = 10

(b) 0

(c) 1

PART – II

SECTION – A

Answer 3.

(i)

A	S	C	L	X
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

POS Expression: Π (0, 1, 2, 3, 4, 5, 7, 8, 9, 12, 13)
: (A + S + C + L) (A + S + C + L′) (A + S + C′ + L) (A + S + C′ + L′) (A + S′ + C + L)
(A +S′ C + L′) (A +S′ + C′ + L′) (A′ + S + C + L) (A′ + S + C + L′) (A′ + S′ + C + L)
(A′ + S′ + C + L′)

(ii) F(A, B, C, D) = Π (0, 1, 2, 3, 4, 5, 7, 8, 9, 12, 13)

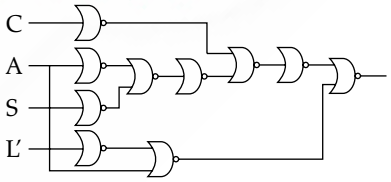
	C + L	C + L′	C′ + L′	C′ + L
A + S	0 1	1 1	3 1	2 1
A + S′	4 1	5 1	7 1	6 0
A′ + S′	12 1	13 1	15 0	14 0
A′ + S	8 1	9 1	11 0	10 0

Octet: C

Quad 1: A+ S

Quad 2: A +L′

Logic gate diagram for the simplified POS expression is: C (A + S) (A + L′)



Answer 4.

(i) (a) F(P, Q, R, S) = Σ(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)

	R′S′	R′S	RS	RS′
P′Q′	0 1	1 1	3 0	2 1
P′Q	4 0	5 1	7 1	6 0
PQ	12 0	13 1	15 1	14 0
PQ′	8 1	9 1	11 0	10 1

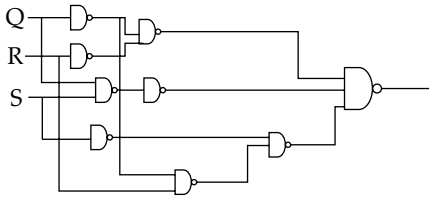
Quad 1: Q'R'

Quad 2: QS

Pair: Q'RS'

(b) Logic gate diagram for the above SOP expression using NAND gates only:

SOP expression: (Q'R')+ (QS) + (Q'RS')



(ii) (a) (1) A' XNOR B'

(2) A OR B'

(3) NOT (A' XNOR B')

(4) NOT (A OR B')

Final Expression: ((NOT (A' XNOR B)) NAND (NOT (A OR B')))

NOT (A'XNOR B) = NOT (A'B+AB')

= NOT (A'B). NOT (AB') (Applying DeMorgan's Theorem)

= (A+B'). (A'+B) (Applying DeMorgan's Theorem) ----- (1)

NOT (A OR B') = A'B ---------- (2)

Combine (1) and (2) using NAND

(A+B') · (A'+B) NAND (A'B)

= (AA'+AB+B'A'+B'B)A'B

= (AB+B'A') A'B (Since AA' and BB' = 0)

= ABA'B+A'B'A'B (Since ABA' = 0)

= A'B'A'B

= A'A'B'B

= A'B' (Since A'A' = A' and B'B = B')

= NOT (A'B') = A+B

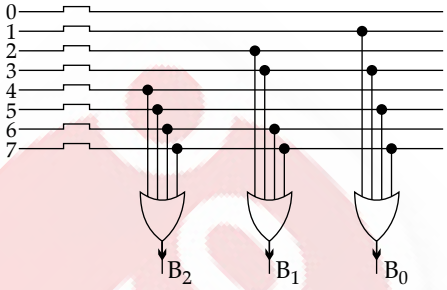
Final simplified expression is: A+B

(b) OR gate

Answer 5.

(i) Encoder: An encoder is a combinational circuit that converts multiple input signals into a coded binary output. It performs the reverse operation of a decoder. For an octal-to-binary encoder, it takes 8 input lines (D0 to D7) and encodes them into a 3-bit binary output (Y2, Y1, Y0).

Logic Circuit Diagram for Octal to Binary Encoder:



Application of Decoder:

Memory Address Decoding

Seven Segment Display (Digital Clocks or Calculators)

Multiplexers

Demultiplexers

Binary to Decimal Conversion

(Any one can be written)

(ii) (~X => Y) ^ X = (X ^ ~Y) V (X ^ Y)

X	Y	~X	~Y	(~X => Y)	(~X => Y) ^ X (1)	X ^ ~Y	X ^ Y	(X ^ ~Y) V (X ^ Y)
0	0	1	1	0	0	0	0	0
0	1	1	0	0	0	0	0	0
1	0	0	1	1	1	1	0	1
1	1	0	0	1	1	0	1	1

(1) and (2) were equal.

Hence, this proposition is a valid proposition.

(iii) Above mentioned logic gate is an XOR gate,

Output = 1

Output = 0

SECTION – B

Answer 6.

```
import java.util.Scanner;
class Perni
{
  private String num; // Binary number as a string
  Perni()
  {
    num = "";
  }
  void accept()
  {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a binary number: ");
    num = sc.next();
    if (!num.matches("[01]+"))
    {
      System.out.println("Invalid input! Enter a
      binary number containing only 0's and 1's.");
    System.exit(0);
    }
}
```

```
int countOne(String binary) {
   int count = 0;
   for (char c : binary.toCharArray()) {
      if (c == '1') count++;
   }
   return count;
}
    void check()
   {
      int decimalNum = Integer.parseInt(num, 2); // Convert binary to decimal
      int onesCount = countOne( num);
      if (isPrime(onesCount))
      {
      System.out.println(num + " is a pernicious number.");
      }
      else
      {
      System.out.println(num + " is NOT a pernicious number.");
      }
   }
   boolean isPrime(int n)
   {
      if (n < 2)
         return false;
      for (int i = 2; i * i <= n; i++)
      {
      if (n % i == 0)
         return false;
      }
      return true;
   }
   public static void main(String[] args)
   {
      Perni obj = new Perni();
      obj.accept();
      obj.check();
   }
}
```

Answer 7.

```
import java.util.Scanner;
class Colsum
{
   private int[][] mat;
   private int m, n;
   public Colsum(int mm, int nn)
   {
      m = mm;
      n = nn;
      mat = new int[m][n];
   }
   public void readArray()
   {
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter elements of the matrix:");
      for (int i = 0; i < m; i++)
      {
         for (int j = 0; j < n; j++)
         {
            mat[i][j] = sc.nextInt();
         }
      }
   }
   public void print()
   {
      System.out.println("Matrix:");
      for (int i = 0; i < m; i++)
      {
         for (int j = 0; j < n; j++)
         {
            System.out.print(mat[i][j] + " ");
         }
         System.out.println();
      }
   }
   public static boolean check(Colsum A, Colsum B)
   {
      for (int j = 0; j < A.n; j++)
      {
         int sumA = 0, sumB = 0;
         for (int i = 0; i < A.m; i++)
         {
            sumA += A.mat[i][j];
            sumB += B.mat[i][j];
         }
         if (sumA != sumB)
         {
            return false;
         }
      }
      return true;
   }
   public static void main(String[] args)
   {
```

```
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int rows = sc.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = sc.nextInt();
        Colsum A = new Colsum(rows, cols);
        Colsum B = new Colsum(rows, cols);
        System.out.println("Enter values for Matrix A:");
        A.readArray();
        System.out.println("Enter values for Matrix B:");
        B.readArray();
        System.out.println("Matrix A:");
        A.print();
        System.out.println("Matrix B:");
        B.print();
        if (Colsum.check(A, B))
        {
          System.out.println("Sum of corresponding columns is equal.");
        }
        else
        {
          System.out.println("Sum of corresponding columns is not equal.");
        }
      }
    }
```

Answer 8.

```
import java.util.Scanner;
class Flipgram
{
  private String word;
  public Flipgram(String s)
  {
    this.word = s;
  }
  public boolean ishetero()
  {
    int len = word.length();
    for (int i = 0; i < len; i++)
    {
      for (int j = i + 1; j < len; j++)
      {
        if (word.charAt(i) == word.charAt(j))
        {
          return false; // Duplicate character found
        }
      }
    }
    return true; // All characters are unique
  }
  public String flip()
  {
    int len = word.length();
    int mid = len / 2;
    if (len % 2 == 0)
    {
      return word.substring(mid) + word.substring(0, mid);
    }
    else
    {
      return word.substring(mid + 1) + word.charAt(mid) + word.substring(0, mid);
    }
  }
  public void display()
  {
    if (ishetero())
    {
      System.out.println("HETEROGRAM");
    }
    else
    {
      System.out.println("Flipped word: " + flip());
    }
  }
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter a word: ");
    String inputWord = sc.next();
    Flipgram obj = new Flipgram(inputWord);
    obj.display();
  }
}
```

SECTION – C

Answer 9.

(i)

```
class CirQueue
{
  private int[] Q;
  private int cap;
  private int front;
  private int rear;
  public CirQueue(int n)
  {
```

```
    cap = n;
    Q = new int[cap];
    front = -1;
    rear = -1;
  }
  public void push(int v)
  {
    if ((rear + 1) % cap == front)
    {
      System.out.println("QUEUE IS FULL");
      return;
    }
    if (front == -1)
    {
      front = 0;
    }
    rear = (rear + 1) % cap;
    Q[rear] = v;
  }
  public int remove()
  {
    if (front == -1)
    {
      return -999;
    }
    int removedValue = Q[front];
    if (front == rear)
    {
      front = -1;
      rear = -1;
    }
    else
    {
      front = (front + 1) % cap;
    }
    return removedValue;
  }
  public void print()
  {
    if (front == -1)
    {
      System.out.println("QUEUE IS EMPTY");
      return;
    }
    System.out.print("Queue elements: ");
    int i = front;
    while (true)
    {
      System.out.print(Q[i] + " ");
      if (i == rear)
      {
        break;
      }
      i = (i + 1) % cap;
      }
      System.out.println();
    }
  }
}
```

(ii) Applications of Circular Queue:

- CPU Scheduling
- Disk Scheduling
- Memory Management
- Printer Queue **(Any one can be written)**

Answer 10.

```
class Passenger extends Flight
{
  private String id;
  private String name;
  private double tax;
  private double tot;
  public Passenger(String flightno, String dep_time,
  String arr_time, double basefare,
  String id, String name)
  {
    super(flightno, dep_time, arr_time, basefare);
    this.id = id;
    this.name = name;
    this.tax = 0.0;
    this.tot = 0.0;
  }
  public void cal()
  {
    tax = 0.05 * basefare;
    tot = basefare + tax;
  }
  public void show()
  {
    super.show(); // Display flight details
    System.out.println("Passenger ID: " + id);
    System.out.println("Passenger Name: " +
    name);
    System.out.println("Tax: " + tax);
    System.out.println("Total Amount to be Paid: "
    + tot);
  }
}
```

Answer 11.

(i)
```
public static void lowercase (Cell str)
{
  int sum = 0;
  Cell temp = str;
  while (temp != null)
  {
  if (Character.isLowerCase(temp.m))
    {
    sum += (int) temp.m;
  }
  temp = temp.right;
  }
  System.out.println("Sum of ASCII values of lowercase letters: " + sum);
}
```

(ii) **(a)** In-order traversal of the right subtree: K, F, N, L, C, G, M

(b) Depth of the binary tree: 4

Depth of node E: 2

(c) External nodes of the left subtree: {I, J, E}

Internal nodes of the right subtree: { C,F,G,L}